INSTITUTE FOR DEFENSE ANALYSES

# Maintaining High Assurance in Asynchronous Messaging

Kevin E. Foltz

William R. Simpson

24 October 2015



IDA Non-Standard
NS D-5490
Log: H 15-000428
Copy

INSTITUTE FOR DEFENSE
ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

*The Institute for Defense Analyses is a non-profit corporation that operates three federally funded research and development centers to provide objective analyses of national security issues, particularly those requiring scientific and technical expertise, and conduct related research on other national challenges.*

**About This Publication**

This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-14-D-0001, Task BC-5-2283, "Architecture, Design of Services for Air Force Wide Distributed Systems," for USAF HQ USAF SAF/CIO A6. The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

# Maintaining High Assurance in Asynchronous Messaging

Kevin E. Foltz and William R. Simpson

***Abstract*—Asynchronous messaging is the delivery of a message without waiting for the intended recipient to respond or acknowledge the message. This solution works for distributed systems communication, in which different systems may or may not be available at the same time. Asynchronous messaging solutions often use a message queue that holds messages to be picked up by the recipient. Although communication with the queue can be secured using lower layer protocols, such as Transport Layer Security (TLS), this does not provide end-to-end security for the sender and receiver. The queuing system acts as a man-in-the-middle, negating authentication, integrity, and confidentiality guarantees. End-to-end security for asynchronous messaging must be provided by the asynchronous messaging layer itself. This paper discusses current asynchronous messaging models and proposes methods for providing end-to-end asynchronous messaging security in a high assurance environment.**

***Index Terms*—Asynchronous Communication, Publish Subscribe, IT Security, Encryption, Key Management**

## I. INTRODUCTION

Asynchronous messaging describes communication that take place between one or more applications or systems, in which the sender does not receive feedback from the receiver during transmission of a message. This is in contrast to synchronous communication, in which the sender of a message waits for acknowledgement or a response from the receiver before completing the transmission.

There is no assumption about which layers asynchronous and synchronous communication take place in or how these relate to each other. It is possible to implement synchronous communication using an asynchronous messaging service or using an asynchronous messaging service using synchronous communication channels. In practice, asynchronous messaging often uses an underlying synchronous channel.

A common asynchronous messaging design involves one system placing a message in a message queue and continuing its processing. At the completion of message transmission, the sender does not know when or whether the receiver received it. The message queuing system is responsible for delivering the message to the recipient. Some systems use two or more queues or intermediaries.

Manuscript received July 2, 2015; revised August 28, 2015. This work was supported in part by the U.S. Secretary of the Air Force and the Institute for Defense Analyses (IDA). The publication of this paper does not indicate endorsement by any organization or IDA, nor should the contents be construed as reflecting the official position of these organizations.
Kevin E. Foltz is with the Institute for Defense Analyses. (email: kfoltz@ida.org). William R. Simpson is with the Institute for Defense Analyses, 4850 Mark Center Drive, Alexandria, Virginia 22311 USA and is the corresponding author phone: 703-845-6637, FAX: 703-845-6848 (e-mail: rsimpson@ida.org )

### A. *Some Advantages of Asynchronous Communication*

Asynchronous messaging solves the problem of intermittent connectivity. If the receiving equipment fails or is unavailable, the message remains in a message queue and is delivered after the failure is corrected. This is especially useful for transmission of large data files, in which failures are more likely and retransmissions more costly.

An asynchronous messaging system with built-in intelligence may transform the content and/or format of the message automatically to conform to the receiving system's requirements or needed protocol but still successfully deliver the message to the recipient. This intelligence is used to provide a higher level of understanding of the content, which allows translation into other formats and protocols. Complicated transformations are better suited to asynchronous communication than synchronous communication because they may increase latency and cause connectivity problems or other underlying protocol failures for synchronous systems.

### B. *Some Disadvantages of Asynchronous Communication*

The disadvantages of asynchronous messaging include the additional component of a message broker or transfer agent to ensure the message is received. This may affect both performance and reliability. Another disadvantage is the response time, which may be inconvenient and not consistent with normal dialog communication.

## II. PRIOR WORK

A proliferation of standards for asynchronous messaging has caused interoperability problems, with each major vendor having its own implementations, interface, and management tools. Java EE systems are not interoperable, and Microsoft's MSMQ (Microsoft Message Queuing) does not support Java EE.

A few of the numerous standard protocols used for asynchronous communication are in the table below.

TABLE 1 MESSAGING PORTS

| Port | TCP/ UDP | Messaging Protocol and Description | Status |
|---|---|---|---|
| 18 | TCP and UDP | The **Message Send Protocol (MSP)**, more precisely referred to as Message Send Protocol 2, is an application layer protocol used to send a short message between nodes on a network. Defined in RFC 1312. | Official |
| 99 | TCP | **WIP message** is a work-in-progress message sent from a computer client to a computer server. It is used to update a server with the progress of an item during a manufacturing process. The only known use is in the automotive wiring manufacturing process, but the message structure is generic enough to be used in any manufacturing process. | Unofficial |
| 110 | TCP | **Post Office Protocol v3 (POP3)** is an email retrieval protocol. | Official |

| Port | TCP/ UDP | Messaging Protocol and Description | Status |
|---|---|---|---|
| 119 | TCP | The **Network News Transfer Protocol (NNTP)** is an application protocol used for transporting Usenet news articles (*netnews*) between news servers and for reading and posting articles by end user client applications. Defined in RFC 3977. | Official |
| 143 | TCP | **Internet Message Access Protocol (IMAP)** is a protocol for e-mail retrieval and storage as an alternative to POP. IMAP, unlike POP, specifically allows multiple clients simultaneously connected to the same mailbox and through flags stored on the server; different clients accessing the same mailbox at the same or different times can detect state changes made by other clients. Defined in RFC 3501. | Official |
| 161 | UDP | **Simple Network Management Protocol (SNMP)** is an "Internet-standard protocol for managing devices on IP networks." Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks, and more. Defined in RFC 3411-3418. | Official |
| 162 | TCP and UDP | **Simple Network Management Protocol Trap (SNMPTRAP).** See port 161. | Official |
| 218 | TCP and UDP | **Message Posting Protocol (MPP)** is a network protocol used for posting messages from a computer to a mail service host. | Official |
| 220 | TCP and UDP | Internet Message Access Protocol (IMAP), version 3. See port 143. | Official |
| 319 | UDP | Event Messages for The **Precision Time Protocol (PTP)** is a protocol used to synchronize clocks throughout a computer network. On a local area network, it achieves clock accuracy in the sub-microsecond range, making it suitable for measurement and control systems. Defined in **IEEE 1588-2008.** | Official |
| 320 | UDP | Event Messages for The **Precision Time Protocol (PTP)**. See port 319. | Official |
| 433 | TCP and UDP | **Network News Speed Protocol (NNSP)**, part of Network News Transfer Protocol for bulk transfer. Defined in RFC 3977. | Official |
| 587 | TCP | **Simple Mail Transfer Protocol (SMTP)**, as specified in RFC 6409. | Official |
| 1801 | TCP and UDP | **Microsoft Message Queuing** or **MSMQ** is a message queue implementation developed by Microsoft and deployed in its Windows Server operating systems since Windows NT 4 and Windows 95. The latest Windows 8 also includes this component. In addition to its mainstream server platform support, MSMQ has been incorporated into Microsoft Embedded platforms since 1999 and the release of Windows CE 3.0. | Official |
| 1863 | TCP | **MSNP (Microsoft Notification Protocol)**, used by the Microsoft Messenger service and a number of Instant Messaging clients. | Official |
| 1935 | TCP | Adobe Systems Macromedia Flash Real Time Messaging Protocol (RTMP) "plain" protocol. | Official |
| 2195 | TCP | Apple Push Notification service Link. | Unofficial |
| 2196 | TCP | Apple Push Notification—Feedback Link. | Unofficial |
| 2948 | TCP and UDP | **Multimedia Messaging Service (MMS)** is a standard way to send messages that include multimedia content to and from mobile phones. It extends the core SMS (Short Message Service) capability that allowed exchange of text messages only up to 160 characters in length. Multimedia Messaging Service 1.3 – Open Mobile Alliance. | Official |
| 2949 | TCP and UDP | WAP-push secure **Multimedia Messaging Service (MMS)**. **S**ee port 2948 Multimedia Messaging Service 1.3 – Open Mobile Alliance. | Official |
| 4486 | TCP and UDP | **Integrated Client Message Service (ICMS).** Defined in RFC 6335. | Official |
| 5010 | TCP | **IBM WebSphere MQ Workflow.** | Official |

### A. *Java Standard Messaging Protocol*

Java Messaging System (JMS) is a message-oriented middleware API for communication between Java clients. It is part of the Java Platform Enterprise Edition. It supports point-to-point communication as well as publish-subscribe.

### B. *De-facto Standard Microsoft Message Queuing*

Microsoft Message Queuing (MSMQ) allows applications running on separate servers/processes to communicate in a failsafe manner. A queue is a temporary storage location from which messages can be sent and received reliably, as and when conditions permit. This enables communication across networks and between computers running Windows, which may not always be connected. By contrast, sockets and other network protocols require permanent direct connections.

MSMQ is responsible for reliably delivering messages between applications inside and outside the enterprise. MSMQ ensures reliable delivery by placing messages that fail to reach their intended destination in a queue and then resending them once the destination is reachable. It also supports security and priority-based messaging. Dead letter queues can be created for looking at messages that have timed out or failed for other reasons.

MSMQ also supports transactions. It permits multiple operations on multiple queues, with all of the operations wrapped in a single transaction, thus ensuring that either all or none of the operations will take effect. Microsoft Distributed Transaction Coordinator (MSDTC) supports transactional access to MSMQ and other resources.

### C. *Open Source Messaging Protocols*

In addition to Java and Microsoft, different open source solutions exist. RabbitMQ is an open source messaging solution that runs on multiple platforms and multiple languages. It implements Advanced Message Queueing Protocol (AMQP), in which messages are queued on a central node before being sent to clients. It is easy to deploy, but having all traffic pass through a single central node can hinder scalability.

ZeroMQ is another cross-platform, cross-language messaging solution that can use different carrier protocols to send messages. It can support publish-subscribe, push-pull, and router-dealer communication patterns. It can be more difficult to set up, but it provides more control and granularity at the lower levels to tune performance.

ActiveMQ is a compromise between the ease of use of Rabbit MQ and the performance of ZeroMQ. All three support multiple platforms and have client APIs for C++, Java, .Net, Python, and others. They also have documentation and active community support. There are many other implementations, including Sparrow, Starling, Kestrel, Beanstalkd, Amazon Simple Queue Service (SQS), Kafka, Eagle MQ, and IronMQ.

### D. *Emerging Standard Advanced Message Queuing Protocol*

Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware. It is an emerging technology addressing the standardization problem. Implementations are interoperable. It includes flexible routing and common message paradigms like publish/subscribe, point-to-point, request-response, and fan-out.

The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability, and security. AMQP mandates the behavior of the messaging provider and client to the extent that implementations from different vendors are truly

interoperable, in the same way as SMTP, HTTP, FTP, and others have created interoperable systems.

## III. ASYNCHRONOUS MESSAGING SECURITY

Asynchronous messaging can provide authentication of the sender and receiver identities and the integrity and confidentiality of the message content if the holder of the queue is trusted. One key challenge in asynchronous messaging systems is that a third party is often involved in the transaction, which may or may not be trusted to speak for the sending or receiving entities or view or modify content in transit. As a result, security models often require a trusted third party, which restricts deployment options. In contrast, synchronous web traffic relies on routers and other infrastructure to deliver messages, but the use of TLS provides end-to-end security without the need to trust these intermediate nodes.

### A. *Security for Server Brokered Invocation*

Server brokered invocation uses web server middleware to manage message queues. The sender and receiver both communicate directly through secure synchronous channels to the server to send and receive messages. This model is shown in Figure 1. Asynchronous message security must be from sender to receiver, not just from sender to server and server to receiver. The latter fails to provide end-to-end authentication, integrity, and confidentiality, which are required for a high assurance environment.

In order for the parties involved in the transaction to provide accountability, integrity, and confidentiality, the service requester must authenticate itself to the receiver, encrypt the message so only the service provider can receive this message, and provide verifiable integrity checks on the full message content. The service provider must confirm that the message is from a known identity, decrypt the content with a valid key, and verify the integrity checks before that entity can take action on the message.

This is accomplished by invoking two cryptographic techniques. The first is the use of a digital signature by the sender. When the message signature is verified, the service provider knows the identity of the sender and that the content has not been altered by another entity after it was signed. The second is the encryption of the message using the public key of the service provider. This requires that the requester know the public key of the target. A response to the requester must similarly be signed and encrypted using the public key of the requester.

The use of asymmetric encryption is paired with more efficient symmetric encryption, where content is encrypted with a random symmetric key, which is itself encrypted using the receiver's public key. Additional security can be provided by message expiration deadlines within queues and central auditing of all messages sent and received.

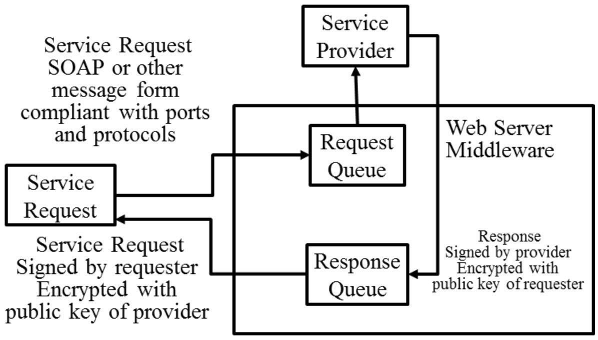


Fig. 1. Security Considerations for Server Brokered Invocation

### B. *Security for Publish Subscribe Systems*

In a Publish Subscribe System (PSS) the queue server acts as an intermediary between sender and receiver to manage many-to-many instead of just many-to-one communications. Senders and receivers communicate with the PSS through a secure synchronous channel. The PSS collects messages and makes them available to entities based on subscriptions. This model is shown in Figure 2.

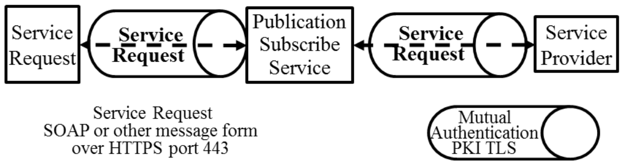


Fig. 2. Publish-Subscribe Push Model

The PSS is an active entity and registered in the Enterprise Service Directory. Active entities act on their own behalf and are not a proxy. To preserve the end-to-end accountability chain for messages, the original publisher signs the message. However, unlike server-brokered invocation, no single public key can be used for all potential receivers. One solution to address this is for the PSS to encrypt the content to the receivers. The sender's signature remains intact, preserving integrity, but end-to-end confidentiality is not guaranteed.

A PSS may use the web server broker as shown in Figure 3. The web server broker is used only for notification messages, so it does not require security like the main channel. The transmission of the actual message is still done through the secure synchronous channel. The storage queue must be encrypted using the PSS's public key. This is piecemeal confidentiality, because the sender encrypts to the PSS, and the PSS encrypts to the receiver. This relies on trust of the PSS.

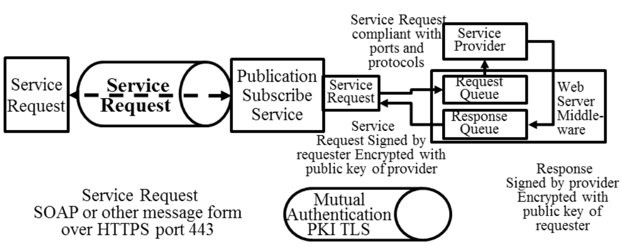


Fig. 3. Publish-Subscribe Pull Model

## IV. PSS ROCK AND JEWEL

The following is an approach developed to maintain high security assurances with the use of an untrusted PSS. In this

formulation, the sender and receiver maintain end-to-end security because the PSS is unable to impersonate either endpoint or view or modify the content. The key concepts are the use of "rocks" and "jewels" to provide security guarantees. The "rocks" are encrypted content blocks, and the "jewels" are the decryption keys for these rocks, encrypted using public keys for the intended recipients.

### A. *Claims for Targeted Content (PSS)*

After authentication through TLS v1.2 or later versions and authorization based on SAML claims, the sender accesses PSS services. The PSS will offer either *publish* or *retrieve* based on the values in the SAML content claim. If there are no SAML content claims, the subscriber will only receive basic services based on identity.

Publishing of content for a targeted list, as used by software publishers, is based upon registered delivery. The targeted list requires the following steps:

0. Publisher does a bi-lateral authentication and establishes a TLS 1.2 session with SAML authorizations for session establishment with the PSS. The PSS identifies him as a publisher. He may also be a subscriber, or he may be modifying a previous publish or he may be retrieving messages, so the PSS ascertains the reason for his session.
1. Content to be published will be digitally signed by the publisher.
2. The publisher will generate an AES-256 encryption key and encrypt the content.
3. Encrypted Content is placed in a queue based on an access claim and list name. The publisher will keep such lists. The PSS will assist in developing claims.
4. Access is based on a list of targets and claims. A target may be an individual subscriber or a group queue. The publisher may establish a new queue based on claims and the list for retrieval. This new queue requires an identity and a claims establishment for retrieval (see 3 above). Additional content may be published as needed.
5. Expiration time of targeted content is determined by the publisher or the messaging system.
6. The PSS will provide PKI certificates for each of the targets for the content (if the publisher needs them and they are already registered in the PSS). The publisher should check all certificates on the list for currency and revocation. If invalid certificates are discovered, the list should be pruned.
7. The publisher will prepare encrypted key sets (jewels) by wrapping the AES encryption key in each target's public key.
8. The publisher will publish the encrypted material (rocks) and the encrypted key sets (jewels) for the targets. The PSS will link these to the encrypted material and the target(s).
9. The PSS will provide notification, if desired, to the subscriber list. The PSS will assist with message selection and target details, or the publisher may script his own.
10. The publisher closes the session.

Note: the target must be on the list and have authorization to view content. The steps are shown in Figure 4.

### B. *Retrieving Content for Known Claimants*

Retrieval of targeted content may be achieved without the targeted identities contacting the publisher. The following steps are followed:

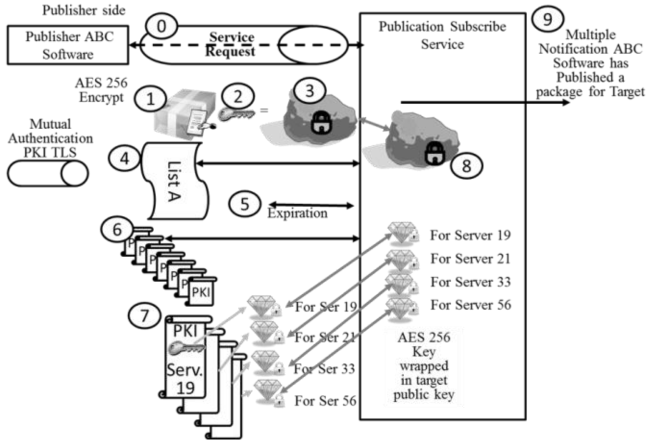


Fig. 4. Publishing of Targeted Content

0. Subscriber does a full bi-lateral authentication using TLS 1.2 with SAML authorizations for session establishment with the PSS. The claims identify him as a subscriber. He may also be a publisher, so the PSS ascertains the reason for his session.
1. The PSS offers subscriber content available for the claims in queues for which the claimant has an encrypted key available, and the subscriber chooses and retrieves the encrypted content (rock).
2. The PSS provides the encrypted key package (jewel).
3. The PSS notifies the publisher. When expiration time occurs, the server deletes the packages and notifies the publisher which packages were not delivered. The publisher may republish to that list if desired.
4. The subscriber decrypts the content encryption key (jewel) with his private key and accesses the content (rock) decryption key.
5. The subscriber decrypts the content.
6. The subscriber verifies and validates signature.
7. The subscriber closes the session or retrieves additional content.

Note: the target must be on the list and have a content claim. The steps are shown in Figure 5.

### C. *Retrieving Content for Unknown Claimants*

Unknown claimants cannot retrieve the content until registering with the content provider. The steps in that process are described below:

0. The subscriber does a full bi-lateral authentication TLS 1.2 with SAML authorizations for session establishment with the PSS. The authentication identifies him as a subscriber. He may also be a publisher, so the PSS ascertains the reason for his session.
1. The PSS checks the content claims available and the subscriber chooses and retrieves the content for which full packages exist.
2. For the unknown list, the encrypted key package is not available. The PSS replies "the publisher has no record of your membership. I need to contact the publisher. I will send you a notice if the publisher agrees."

3. The PSS stores a message for the publisher and notifies him that he has a message.
4. The PSS and subscriber await publisher action.
5. The subscriber closes the session or retrieves additional content.

Note: the target has a content claim, but is not on the list. The steps are shown in the next figure.

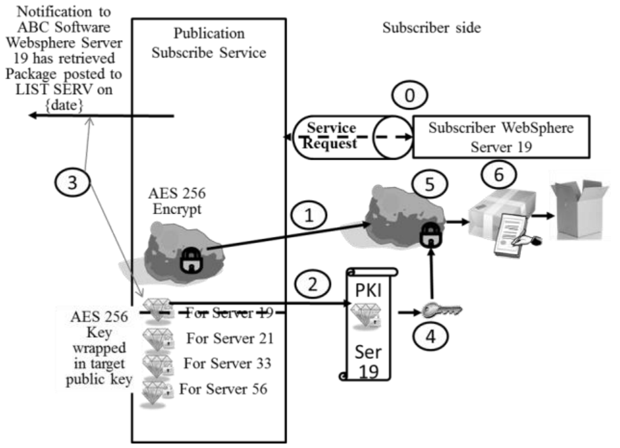


Fig. 5. Subscriber Retrieval(s) from a Known Target

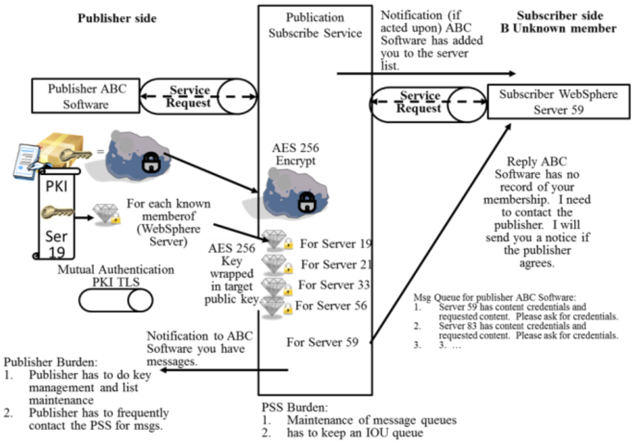


Fig. 6. Subscriber Retrieval(s) from an Unknown Target

## D. *Adjusting Publishing Targets (Untrusted PSS)*

0. The publisher does a full bi-lateral authentication TLS 1.2 with SAML authorizations for session establishment with the PSS. The authorization process identifies him as a publisher. He may also be a subscriber, or he may be modifying a previous publish or he may be retrieving messages, so the PSS ascertains the reason for his session.
   1. Retrieve messages. These are retrieved one by one with action taken (or not) and deletion of the message.
      - The publisher asks for credentials of previously unknown claimants he wishes to add to his lists.
        – The publisher may add claimants to the publisher's list
        – The publisher computes jewels.
        – The publisher posts jewels.
        – The PSS notifies the subscriber that he has content available. This makes the entity a known target and SECTION V B applies.
      - PSS provides messages to requester.
   2. The publisher closes the session.

The steps are shown in Figure 7.

## E. *Distribution of Burdens*

Several burdens are incurred in this high security mode. The publisher has to do key management and list maintenance. The publisher has to frequently contact the PSS for messages for publishers. The PSS must maintain message queues for publishers. The PSS has to keep a linked wrapped key package by target with published content. The PSS is responsible for additional notifications that are sent out. The unknown claimant may have a delay in receiving content to which he has claims.

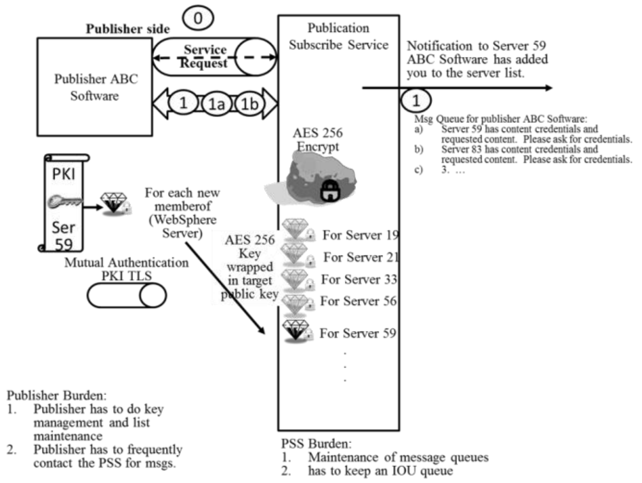


Fig. 7. Publisher Message Retrieval and Subsequent Actions

## V. SUMMARY

We have reviewed the basic approaches to asynchronous communication in computing environments. We have also described high assurance approaches to the process. The proliferation of standards in this area has created a problem with high assurance. In many instances the high assurance elements require additional steps in the asynchronous process, but they provide a way to proceed when some intermediaries are untrusted. This work is part of a body of work for high assurance enterprise computing using web services. Elements of this work are described in [24-37].

## REFERENCES


[1] World Wide Web Consortium (W3C):
  a. XML Encryption Syntax and processing, 10 December 2002.
  b. XML Signature Syntax and Processing (Second Edition), 10 June 2008.
  c. Canonical XML Version 1, March 2001.
  d. Exclusive Canonical XML Version 1, July, 2002.

[2] Organization for the Advancement of Structured Information Standards (OASIS) open set of Standards:
  a. "WS-Security Specification 1.1," OASIS, November 2006.
  b. "WS-Trust Specification 1.4," OASIS, February 2009.
  c. "WS-ReliableMessaging Specification 1.1," OASIS, November 2004.
  d. "WS-SecureConversation Specification 1.4," OASIS, February 2009.
  e. "WS-BaseNotification," 1.3 OASIS, October 2006.
  f. "WS-BrokeredNotification," 1.3 OASIS, October 2006.
  g. N. Ragouzis et al., Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS Committee Draft, March 2008.
  h. P. Mishra et al. Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, March 2005.

i. S. Cantor et al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, March 2005.
a. OASIS Advanced Message Queuing Protocol, (AMQP) Version 1.0, OASIS Standard, 29 October 2012.

[3] Standard for Naming Active Entities on DoD IT Networks, Version 3.5 (or current), Sept. 23, 2010.

[4] National Institute of Standards, Gaithersburg, MD:
a. FIPS PUB 197, Advanced Encryption Standard (AES), November 2001.
b. FIPS PUB 800-67, November 2008.
c. FIPS PUB 800-67, Version 1.2, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Revised July 2011.
d. FIPS PUB 140-2, Security Requirements for Cryptographic Modules, May 25, 2001.
e. FIPS PUB 180-2, Secure Hash Standard. August 2002, U.S. Department of Commerce/National Institute of Standards and Technology.
f. FIPS PUB 186-3, Digital Signature Standard (DSS), June, 2009.
g. FIPS PUB 800-38, Recommendation for Block Cipher Modes of Operation: 38A, Methods and Techniques, December 2001; 38B, The RMAC Authentication Mode, November 5 2002 draft; 38C, The CCM Mode for Authentication and Confidentiality.
h. FIPS PUB 800-53, Recommended Security Controls for Federal Information Systems and Organizations, Revision 3, August 2009.

[5] Internet Engineering Task Force (IETF) Standards:
a. STD 9 (RFC0959) File Transfer Protocol, J. Postel, J. Reynolds, October 1985.
b. STD 5 (RFC0791) Internet Protocol, J. Postel, September 1981, and subsequent RFCs 791/950/919/922/792/1112.
c. STD 66 (RFC3986) Uniform Resource Identifier (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter, January 2005.
d. RFC 1321, "The MD5 Message-Digest Algorithm," April 1992.
e. RFC 2104, HMAC: Keyed-Hashing for Message Authentication, Feb 1997.
f. RFC 2406, IP Encapsulating Security Payload, November 1998.
g. RFC 2459, Internet X.509 PKI – Certificate and CRL Profile, January 1999.
h. RFC 2560, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, June 1999.
i. RFC 2829, Authentication Methods for LDAP. M. Wahl, H. Alvestrand, J. Hodges, R.L. Morgan. May 2000.
j. RFC 4510 Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map, June 2006.
k. RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2," August 2008.
l. RFC 5751 Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification, January 2010.
m. RFC 5751, Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2, July 2010.
n. RFC 6151: "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms," March 2011.

[6] PKCS #1: RSA Cryptography Standard: ASN Module for PKCS #1 v2.1, June 14, 2002.

[7] MSMQ Queuing (MSMQ) http://msdn.microsoft.com/en-us/library/ms711472(v=vs.85).aspx.

[8] DoDI 8500.2 DoD Instruction, Information Assurance (IA) Implementation, 6 February 2003.

[9] Security Technical Implementation Guide (STIG), Version 5, R1, 28 March 2006.

[10] DoD 5200.1-R "Information Security Program," January 1997.

[11] DoD Directive 8320.2, "Data Sharing in a Net-Centric Department of Defense," December 2, 2004, certified current April 23, 2007.

[12] MSMQ Queuing (MSMQ) http://msdn.microsoft.com/en-us/library/ms711472(v=vs.85).aspx.

[13] Enterprise Integration Patterns by Gregor Hohpe and Bobby Woolf, http://eaipatterns.com/index.html.

[14] Why developers do need an Enterprise Service Bus? – http://www.ibm.com/developerworks/webservices/library/ws-whyesb/.

[15] The Message Bus Pattern – http://www.ibm.com/developerworks/webservices/library/ws-tip-altdesign4/.

[16] Eventing in SOA – http://crishantha.com/wp/?p=885.

[17] Message Broker Comparison– http://lifecorporatedev.blogspot.com/2012/07/recently-i-have-been-given-task-to-find.html.

[18] RabbitMQ – http://www.rabbitmq.com/.

[19] Apache ActiveMQ – http://activemq.apache.org/.

[20] Apache Qpid – http://qpid.apache.org/.

[21] JBoss HornetQ – http://www.jboss.org/hornetq.

[22] ZeroMQ – http://www.zeromq.org/.

[23] WebSphereWMQ–http://www-01.ibm.com/software/integration/wmq/.

[24] William R. Simpson, Coimbatore Chandersekaran and Andrew Trice, Electronic Digest of the 2008 System and Software Technology Conference, "A Persona-Based Framework for Flexible Delegation and Least Privilege," Las Vegas, Nevada, May 2008.

[25] William R. Simpson, Coimbatore Chandersekaran and Andrew Trice, The 1st International Multi-Conference on Engineering and Technological Innovation: IMET2008, "Cross-Domain Solutions in an Era of Information Sharing," Volume I, pp.313–318, Orlando, FL, June 2008.

[26] Coimbatore Chandersekaran and William R. Simpson, World Wide Web Consortium (W3C) Workshop on Security Models for Device APIs, "The Case for Bi-lateral End-to-End Strong Authentication," 4 pp., London, England, December 2008.

[27] William R. Simpson and Coimbatore Chandersekaran, The 2nd International Multi-Conf. on Engineering and Technological Innovation: IMETI2009, Volume I, pp. 300–305, "Information Sharing and Federation," Orlando, FL, July 2009.

[28] Coimbatore Chandersekaran and William R. Simpson, The 3rd International Multi-Conf. on Engineering and Technological Innovation: IMETI2010, Volume 2, "A SAML Framework for Delegation, Attribution and Least Privilege," pages 303–308, Orlando, FL, July 2010.

[29] William R. Simpson and Coimbatore Chandersekaran, The 3rd International Multi-Conference on Engineering and Technological Innovation: IMETI2010, Volume 2, "Use Case Based Access Control," pages 297–302, Orlando, FL, July 2010.

[30] Coimbatore Chandersekaran and William R. Simpson, The First International Conference on Computer Science and Information Technology (CCSIT-2011), "A Model for Delegation Based on Authentication and Authorization," Springer Verlag Berlin-Heildleberg, Lecture Notes in Computer Science, 20 pp.

[31] William R. Simpson and Coimbatore Chandersekaran, The 16th International Command and Control Research and Technology Symposium: CCT2011, Volume II, pp. 84–89, "An Agent Based Monitoring System for Web Services," Orlando, FL, April 2011.

[32] William R. Simpson and Coimbatore Chandersekaran, International Journal of Computer Technology and Application (IJCTA), "An Agent-Based Web-Services Monitoring System," Vol. 2, No. 9, September 2011, pp. 675–685.

[33] William R. Simpson, Coimbatore Chandersekaran and Ryan Wagner, Lecture Notes in Engineering and Computer Science, Proceedings World Congress on Engineering and Computer Science 2011, Volume I, "High Assurance Challenges for Cloud Computing," pp. 61–66, San Francisco, October 2011.

[34] Coimbatore Chandersekaran and William R. Simpson, Lecture Notes in Engineering and Computer Science, Proceedings World Congress on Engineering 2012, The 2012 International Conference of Information Security and Internet Engineering, Volume I, "Claims-Based Enterprise-Wide Access Control," pp. 524–529, London, July 2012.

[35] William R. Simpson and Coimbatore Chandersekaran, Lecture Notes in Engineering and Computer Science, Proceedings World Congress on Engineering 2012, The 2012 International Conference of Information Security and Internet Engineering, Volume I, "Assured Content Delivery in the Enterprise," pp. 555–560, London, July 2012.

[36] William R. Simpson and Coimbatore Chandersekaran, Lecture Notes in Engineering and Computer Science, Proceedings World Congress on Engineering and Computer Science 2012, Volume 1, "Enterprise High Assurance Scale-up," pp. 54–59, San Francisco, October 2012.

[37] Coimbatore Chandersekaran and William R. Simpson, International Journal of Scientific Computing, Vol. 6, No. 2, "A Uniform Claims-Based Access Control for the Enterprise," December 2012, ISSN: 0973-578X, pp. 1–23.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| | | |
|---|---|---|
| 1. REPORT DATE (DD-MM-YY)<br>24-10-15 | 2. REPORT TYPE<br>Non-Standard | 3. DATES COVERED (From – To) |

| | |
|---|---|
| 4. TITLE AND SUBTITLE<br>Maintaining High Assurance in Asynchronous Messaging | 5a. CONTRACT NUMBER<br>HQ0034-14-D-0001<br>5b. GRANT NUMBER<br>5c. PROGRAM ELEMENT NUMBERS |
| 6. AUTHOR(S)<br>Kevin E. Foltz, William R. Simpson | 5d. PROJECT NUMBER<br>BC-5-2283<br>5e. TASK NUMBER<br>5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESSES<br>Institute for Defense Analyses<br>4850 Mark Center Drive<br>Alexandria, VA 22311-1882 | 8. PERFORMING ORGANIZATION REPORT NUMBER<br>NS D-5490<br>H 15-000428 |
| 9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>Frank P. Konieczny<br>USAF HQ USAF SAF/CIO A6 | 10. SPONSOR'S / MONITOR'S ACRONYM<br>SAF CIO/CTO<br>11. SPONSOR'S / MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
Project Leader: William R. Simpson

14. ABSTRACT

Asynchronous messaging is the delivery of a message without waiting for the intended recipient to respond or acknowledge the message. This solution works for distributed systems communication, in which different systems may or may not be available at the same time. Asynchronous messaging solutions often use a message queue that holds messages to be picked up by the recipient. Although communication with the queue can be secured using lower layer protocols, such as Transport Layer Security (TLS), this does not provide end-to-end security for the sender and receiver. The queuing system acts as a man-in-the-middle, negating authentication, integrity, and confidentiality guarantees. End-to-end security for asynchronous messaging must be provided by the asynchronous messaging layer itself. This paper discusses current asynchronous messaging models and proposes methods for providing end-to-end asynchronous messaging security in a high assurance environment.

15. SUBJECT TERMS
Asynchronous Communication, Publish Subscribe, IT Security, Encryption, Key Management

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Frank P. Konieczny |
| Unclassified | Unclassified | Unclassified | Unlimited | 6 | 19b. TELEPHONE NUMBER (Include Area Code)<br>(703) 697-1308 |